Tracealyzer using Lauterbach TRACE32 for ITM streaming and snapshot

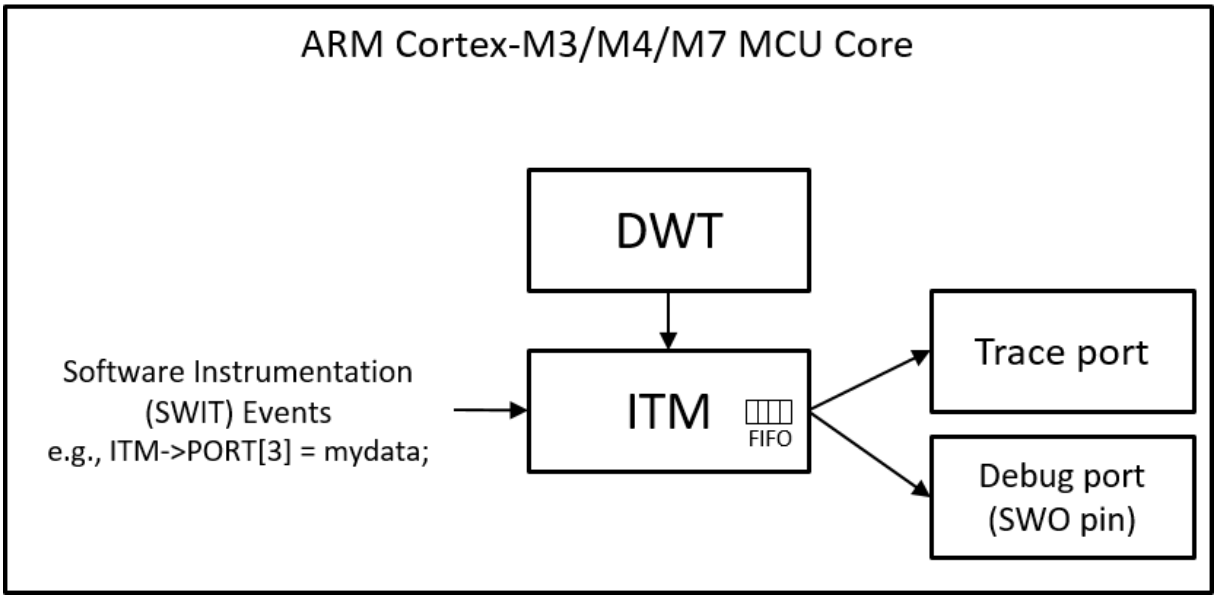
Application Note PA-033 , 2021-02-22

This application note will mostly cover setting up ITM streaming for Tracealyzer when a debugger from Lauterbach is used. At the end there is also a short section on how you can use snapshot mode. The snapshot mode is mostly intended for the use case when ITM/SWO capabilities are not available.

As of version 4.1, Tracealyzer supports ITM tracing for ARM Cortex-M3, M4 and M7 MCUs. This is available for FreeRTOS, Micrium µC/OS-III and SafeRTOS. An example screenshot of Tracealyzer using this setup is shown above.

The ITM interface (short for *Instrumentation Trace Macrocell*) allows the Tracealyzer recorder library to send event data via the debug probe, either using the SWO pin (Serial Wire Output), or 4-bit TPIU trace. SWO is found on all ARM Cortex Debug connectors and is supported by most debug probes.

https://percepio.com



With a fast debug probe ITM allows for high data rates (over 2 MB/s) and the target-side overhead is minimal. In our experiments, writing 16 bytes of data took just 55-60 clock cycles, despite requiring four separate writes. Moreover, unlike some other streaming solutions we support, ITM tracing does not require any RAM buffer.

Requirements for ITM streaming

To follow the instructions in this document, you need the following:

- Target system: Any ARM processor with ITM support.
  - Supported by practically all Arm Cortex-M3, M4, M7 or M33 MCUs.
  - Not supported by ARM Cortex-M0 or M0+ MCUs.

- RTOS: An RTOS supported by Percepio Tracealyzer using the Percepio recorder library
  - FreeRTOS (v7.3 or newer)
  - Micirum µC/OS-III (v3.04 or newer)
  - SafeRTOS

- Development tools
  - TRACE32 debugger software, any version supporting the used CPU could be used
  - Any Lauterbach HW that supports either SWO trace or TPIU trace for Cortex-M: uTrace, Combiprobe or Power Debug Pro + Arm debug cable(v5)
  - Percepio Tracealyzer v4.1 or newer, with a license matching your RTOS.

Target-side setup

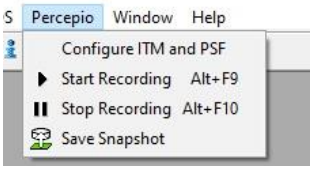
Follow the getting started guide for streaming mode, as provided in the Tracealyzer User Manual ("Integrating the recorder" and "Setting up streaming"). Make sure to include the "stream port" found in "TraceRecorder/streamports/ARM_ITM". This is a generic solution for ITM tracing that works with any IDE and debug probe, assuming they have Trace/SWO support and an ability to output the ITM data to a binary file.

When calling vTraceEnable to initialize the recorder, we recommend using the parameter "TRC_START", which starts the tracing directly. For manual control "TRC_START_AWAIT_HOST" could also be used.

We recommend using ITM stimulus port 1, but it could be modified in trcStreamingPort.h. If this is modified the change must also be reflected in the debugger configuration in the next step.

Debugger setup

Some TRACE32 scripts are available in TraceRecorder/streamports/ARM_ITM, they are also attached to this application note. It is recommended to copy these scripts to your working directory and run the script percepio.cmm to create an extra menu in TRACE32.



First of all, you need to configure your debugger for SWO or TPIU trace. Notice that sometimes the name Serial Wire Viewer (SWV) is used instead of SWO, but it basically refers to the same functionality. How to set this up is outside the scope of this application note, but below you can see a short snippet of a script to configure SWO trace. For more details check the Lauterbach uTrace for Cortex-M User's Guide.

```
55  TPIU.PortSize SWV
56  TPIU.PortMode NRZ
57  TPIU.SWVPrescaler 1.   ;With uTrace+high speed whisker 168Mbit/s works fine
58  CAnalyzer.AutoFocus
59
```

Next you need to enable the ITM port and configure the psf file used as a pipe to Tracealyzer. If default settings should be used this can be configured by using the menu "Percepio/Configure

ITM and PSF". If you would like to change the ITM port number or the location of the psf file you could call the script with parameters:

```
do itm_streaming.cmm <trace file> <ITM channel.>
```

You could also add a call to the script with or without parameters from your normal startup script. In addition to enabling the ITM stimulus port, the script configures the ITM so that all stimulus ports are accessible from user code. This could be modified according to your needs.

Recording and viewing traces in Tracealyzer – Basic Approach

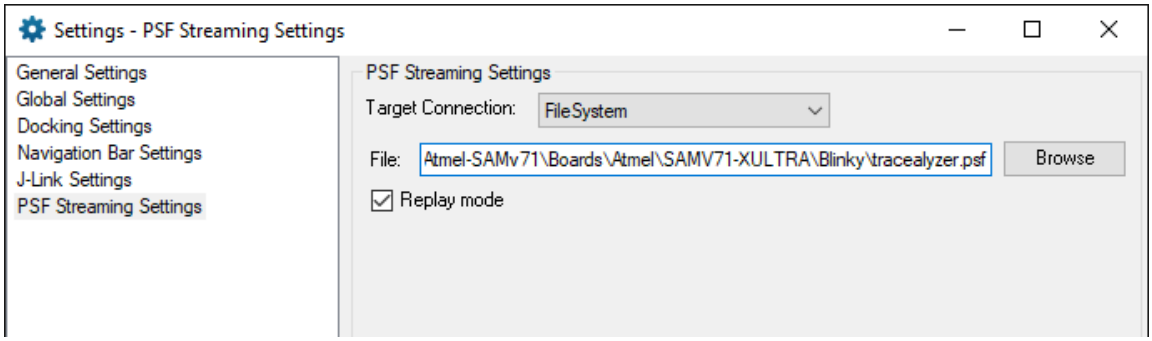
Once you have integrated the trace recorder in your project, configured it for streaming using the ARM_ITM stream port, and configured TRACE32 as instructed above, you can start recording traces and view them in Tracealyzer. The most basic approach is:

1. Start a new debug session with the debugger configuration described in previous section and let your system run. The tracing starts directly if you use "TRC_START" in your call to vTraceEnable. If you use "TRC_START_AWAIT_HOST" then you could use the menu "Percepio/Start Recording" to start tracing.
2. When you have recorded your test session, use menu "Percepio/Stop Recording" or just halt execution of the CPU with the debugger.
3. You should now have a trace file, "tracealyzer.psf", in your working directory or in the location you specified if you didn't use the default settings. Open this file in Tracealyzer using File -> Open, or simply drag the file into Tracealyzer.

Note that the next trace session will overwrite the tracealyzer.psf file, so if you want to save your trace you need to move or rename this file.

Recording and viewing traces in Tracealyzer – File Streaming and Live Visualization

Tracealyzer 4 supports live visualization in several views. You may view ITM traces collected by TRACE32 live, by configuring Tracealyzer to stream the data from the trace file produced by TRACE32. Tracealyzer can read and show the trace file "live" while it is being recorded.
Open the Tracealyzer settings (File -> Settings, or "Recorder Settings" on the start screen) and select PSF streaming settings. Set Target Connection to "FileSystem" and select "Browse" to specify your tracealyzer.psf file. Note that Tracealyzer regards this as a temporary file, just a "data source", and still saves the trace data in a separate permanent file. So you don't need to make copies of tracealyzer.psf manually.



The “Replay mode” decides how Tracealyzer reads the data, which affects the smoothness and latency of the display.

- Replay mode enabled: This mode limits the speed of Tracealyzer to match the target system timestamps. As long as there is some data in the file when Tracealyzer begins to read, it typically won’t run out of data to display. This way you get smooth live views. However, you will also see some additional latency between the target system and the Tracealyzer views.

- Replay mode disabled: Displays the data as fast as possible, in the rate it arrives in the file. When using this for TRACE32 ITM streaming, you minimize the latency between target system and Tracealyzer views, but the live views will not update as smoothly as in replay mode.

To view the trace live in Tracealyzer, follow these steps:

1. Start a new debug session with all the necessary configurations for ITM streaming and start execution of the CPU.
2. Select “Start Recording” in Tracealyzer. Tracealyzer now goes into live mode and begins displaying the trace. Note that some views and features are not available in live mode.
3. When satisfied, select “Stop Recording” in Tracealyzer to end the live mode and enable all views and features in Tracealyzer.

Snapshot mode

If your system does not support SWO or TPIU trace, you can't use streaming but you still have the option to use snapshot mode. The target-side need to be configured with
#define TRC_CFG_RECORDER_MODE TRC_RECORDER_MODE_SNAPSHOT
in trcConfig.h, you find more information about this in the user manual.
If you want to create a snapshot, use the menu "Percepio/Save Snapshot" or the script
do save_snapshot.cmm mytrace2.bin
Without parameter, the default file name will be tracealyzer.bin.
Use "Open File" or drag and drop in Tracealyzer to load the trace.

Additional Notes

- During tracing, you may stop on breakpoints, single-step and resume execution without interfering with the resulting trace. This is possible because the event timestamps are set based on the target system cycle counter, so halting the system "stops the time" from the perspective of the recorder library. However, like always, halting the system may interfere with the target system as a whole, e.g. if this is controlling a physical system with real-time requirements.

- If you run code in user mode (i.e. with memory protection) you should also check that the ITM privilege flags is set accordingly. Otherwise you are not allowed to write to the ITM port from restricted user mode code, e.g. using vTracePrintF.

- Do. not enable cycle accurate trace for the ITM. The Tracealyzer recorder library provides timestamps as part of the event data and allows for Exception (ISR) tracing via explicit logging calls in the ISR handlers (vTraceStoreISRBegin and vTraceStoreISREnd, described in trcRecorder.h and in the Tracealyzer User Manual).

- If you do multiple traced runs within the same debug session, they will be appended to the same tracealyzer.psf file unless this file is reset between trace sessions. Appending multiple traces to the same file is not suitable in replay mode, as the file is replayed from the beginning when starting a new Tracealyzer session.

    - You may make multiple (separate) traces within the same debug session, by following these steps to reset the trace file:

      ```
      do stop_recording.cmm
      ```

      Stop recording in Tracealyzer

      ```
      Trace.WRITE
      Trace.WRITE tracealyzer.psf /ChannelID 1. /Payload
      do start_recording.cmm
      ```

      Start recording in Tracealyzer

Learning more

To learn more about Tracealyzer and ARM ITM, we recommend the following resources:

- Tracealyzer User Manual (Help -> User Manual)
- Tracealyzer “Getting Started” portal - https://percepio.com/gettingstarted
- About ITM trace - https://percepio.com/2016/06/09/arm-itm/
- About the recorder and custom streaming, https://percepio.com/2016/10/05/rtos-tracing

The cmm scripts mentioned in this Application Note can be downloaded here.

For questions, please contact support@percepio.com